

Efficient and Optimal Binary Hopfield Associative Memory Storage Using Minimum Probability Flow

Christopher Hillar*

Mathematical Sciences Research Institute
Berkeley, CA 94720 and
Redwood Center for Theoretical Neuroscience
University of California, Berkeley
Berkeley, CA 94720

Jascha Sohl-Dickstein[†] and Kilian Koepsell[‡]

Redwood Center for Theoretical Neuroscience
University of California, Berkeley
Berkeley, CA 94720

We present an algorithm to store binary memories in a Hopfield neural network using minimum probability flow, a recent technique to fit parameters in energy-based probabilistic models. In the case of memories without noise, our algorithm provably achieves optimal pattern storage (which we show is at least one pattern per neuron) and outperforms classical methods both in speed and memory recovery. Moreover, when trained on noisy or corrupted versions of a fixed set of binary patterns, our algorithm finds networks which correctly store the originals. We also demonstrate this finding visually with the unsupervised storage and clean-up of large binary fingerprint images from significantly corrupted samples.

Introduction. In 1982, motivated by the Ising spin glass model from statistical physics [1], Hopfield introduced an auto-associative neural-network for the storage and retrieval of binary patterns [2]. Even today, this model and its various extensions [3, 4] provide a plausible mechanism for memory formation in the brain. However, existing techniques for training Hopfield networks suffer either from limited pattern capacity or excessive training time, and they exhibit poor performance when trained on unlabeled, corrupted memories.

Our main theoretical contributions here are the introduction of a tractable and neurally-plausible algorithm for the optimal storage of patterns in a Hopfield network, a proof that the capacity of such a network is at least one pattern per neuron, and a novel local learning rule for training neural networks. Our approach is inspired by minimum probability flow [5], a recent technique for fitting probabilistic models that avoids computations with a partition function, the usually intractable normalization constant of a parameterized probability distribution.

We also present several experimental results. When compared with standard techniques for Hopfield pattern storage, our method is shown to be superior in efficiency and generalization. Another finding is that our algorithm can store many patterns in a Hopfield network from highly corrupted (unlabeled) samples of them. This discovery is also corroborated visually by the storage of 64×64 binary images of human fingerprints from highly corrupted versions, as explained in Fig. 2.

Background. A Hopfield network $\mathcal{H} = (\mathbf{J}, \theta)$ on n nodes $\{1, \dots, n\}$ consists of a symmetric weight matrix $\mathbf{J} = \mathbf{J}^\top \in \mathbb{R}^{n \times n}$ with zero diagonal and a threshold vector $\theta = (\theta_1, \dots, \theta_n)^\top \in \mathbb{R}^n$. The possible states of the

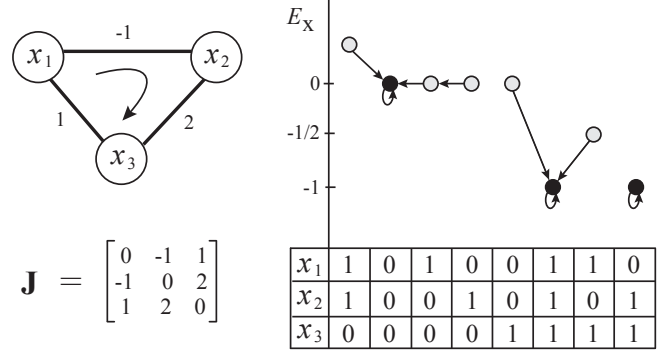


FIG. 1. Example Hopfield Network. The figure above displays a 3-node Hopfield network with weight matrix \mathbf{J} and zero threshold vector. Each binary state vector $\mathbf{x} = (x_1, x_2, x_3)^\top$ has energy $E_{\mathbf{x}}$ as labeled on the y -axis of the diagram on the right. Arrows between states represent one iteration of the network dynamics; i.e., x_1, x_2 , and x_3 are updated by (1) in the order indicated by the clockwise arrow in the graph on the left. The resulting fixed states of the network are indicated by filled circles.

network are all length n binary strings $\{0, 1\}^n$, which we represent as binary column vectors $\mathbf{x} = (x_1, \dots, x_n)^\top$, each $x_i \in \{0, 1\}$ indicating the state x_i of node i . Given any state $\mathbf{x} = (x_1, \dots, x_n)^\top$, an (asynchronous) *dynamical update* of \mathbf{x} consists of replacing x_i in \mathbf{x} (in consecutive order starting with $i = 1$; see Fig 1) with the value

$$x_i = H(\mathbf{J}_i \mathbf{x} - \theta_i). \quad (1)$$

Here, \mathbf{J}_i is the i th row of \mathbf{J} and H is the Heaviside function given by $H(r) = 1$ if $r > 0$ and $H(r) = 0$ if $r \leq 0$.

The energy $E_{\mathbf{x}}$ of a binary pattern \mathbf{x} in a Hopfield

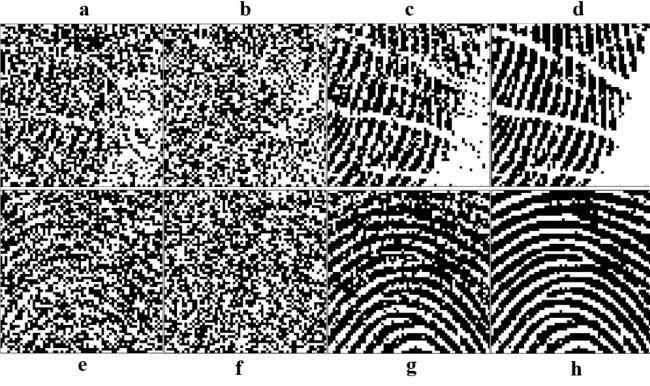


FIG. 2. **Learning memories from corrupted samples.** We stored 80 fingerprints (64×64 binary images) in a Hopfield network with $n = 64^2 = 4096$ nodes by minimizing the MPF objective (4) over a large set of randomly generated (and unlabeled) “noisy” versions (each training pattern had a random subset of 1228 of its bits flipped; e.g., a,e). After training, all 80 original fingerprints were stored as fixed-points of the network. **a.** Sample fingerprint with 30% corruption used for training. **b.** Sample fingerprint with 40% corruption. **c.** State of the network after one update of the dynamics initialized at b. **d.** Converged network dynamics equal to original fingerprint. **e-h.** As in a-d, but for a different fingerprint.

network is defined to be

$$E_{\mathbf{x}}(\mathbf{J}, \theta) := -\frac{1}{2} \mathbf{x}^\top \mathbf{J} \mathbf{x} + \theta^\top \mathbf{x} = -\sum_{i < j} x_i x_j J_{ij} + \sum_{i=1}^n \theta_i x_i, \quad (2)$$

identical to the energy function for an Ising spin glass. In fact, the dynamics of a Hopfield network can be seen as 0-temperature Gibbs sampling of this energy function. A fundamental property of Hopfield networks is that asynchronous dynamical updates do not increase the energy (2). Thus, after a finite number of updates, each initial state \mathbf{x} converges to a *fixed-point* $\mathbf{x}^* = (x_1^*, \dots, x_n^*)^\top$ of the dynamics; that is, $x_i^* = H(\mathbf{J}_i \mathbf{x}^* - \theta_i)$ for each i . See Fig. 2 for a sample Hopfield network on $n = 3$ nodes.

Given a binary pattern \mathbf{x} , the *neighborhood* $\mathcal{N}(\mathbf{x})$ of \mathbf{x} consists of those binary vectors which are Hamming distance 1 away from \mathbf{x} (i.e., those with exactly one bit different from \mathbf{x}). We say that \mathbf{x} is a *strict local minimum* if every $\mathbf{x}' \in \mathcal{N}(\mathbf{x})$ has a strictly larger energy:

$$0 > E_{\mathbf{x}} - E_{\mathbf{x}'} = (\mathbf{J}_i \mathbf{x} - \theta_i) \delta_i, \quad (3)$$

where $\delta_i = 1 - 2x_i$ and x_i is the bit that differs between \mathbf{x} and \mathbf{x}' . It is straightforward to verify that if \mathbf{x} is a strict local minimum, then it is a fixed-point of the dynamics.

A basic problem is to construct Hopfield networks with a given set \mathcal{D} of binary patterns as fixed-points or strict local minima of the energy function (2). Such networks are useful for memory denoising and retrieval since corrupted versions of patterns in \mathcal{D} will converge through the dynamics to the originals. Traditional approaches

to this problem consist of iterating over \mathcal{D} a *learning rule* [6] that updates a network’s weights and thresholds given a training pattern $\mathbf{x} \in \mathcal{D}$. We call a rule *local* when the learning updates to the three parameters J_{ij} , θ_i , and θ_j can be computed with access solely to x_i, x_j , the feedforward inputs $\mathbf{J}_i \mathbf{x}$, $\mathbf{J}_j \mathbf{x}$, and the thresholds θ_i, θ_j ; otherwise, we call the rule *nonlocal*. The locality of a rule is an important feature in a network training algorithm because of its necessity in theoretical models of computation in neuroscience.

In [2], Hopfield defined an *outer-product learning rule* (OPR) for finding such networks. OPR is a local rule since only the binary states of nodes x_i and x_j are required to update a coupling term J_{ij} during training (and only the state of x_i is required to update θ_i). Using OPR, at most $n/(4 \log n)$ patterns can be stored without errors in an n -node Hopfield network [7, 8]. In particular, the ratio of patterns storable to the number of nodes using this rule is at most $1/(4 \log n)$ memories per neuron, which approaches zero as n increases. If a small percentage of incorrect bits is tolerated, then approximately $0.15n$ patterns can be stored [2, 9].

The *perceptron learning rule* (PER) [10, 11] provides an alternative method to store patterns in a Hopfield network [12]. PER is also a local rule since updating J_{ij} requires only $\mathbf{J}_i \mathbf{x}$ and $\mathbf{J}_j \mathbf{x}$ (and updating θ_i requires $\mathbf{J}_i \mathbf{x}$). Unlike OPR, it achieves optimal storage capacity, in that if it is possible for a collection of patterns \mathcal{D} to be fixed-points of a Hopfield network, then PER will converge to parameters \mathbf{J}, θ for which all of \mathcal{D} are fixed-points. However, training frequently takes many parameter update steps (see Fig. 4), and the resulting Hopfield networks do not generalize well (see Fig. 5) nor store patterns from corrupted samples (see Fig. 6).

Despite the connection to the Ising model energy function, and the common usage of Ising spin glasses (otherwise referred to as Boltzmann machines [4]) to build probabilistic models of binary data, we are aware of no existing work on associative memories that takes advantage of a probabilistic interpretation during training. (Although probabilistic interpretations have been used for pattern recovery [13].)

Theoretical Results. We give an efficient algorithm for storing at least n binary patterns as strict local minima (and thus fixed-points) in an n -node Hopfield network, and we prove that this algorithm achieves the optimal storage capacity achievable in such a network. We also present a novel local learning rule for the training of neural networks.

Consider a collection of m binary n -bit patterns \mathcal{D} to be stored as strict local minima in a Hopfield network. Not all collections of m such patterns \mathcal{D} can so be stored; for instance, from (3) we see that no two binary patterns one bit apart can be stored simultaneously. Nevertheless, we say that the collection \mathcal{D} *can be stored as local minima* of a Hopfield network if there is some $\mathcal{H} = (\mathbf{J}, \theta)$ such

that each $\mathbf{x} \in \mathcal{D}$ is a strict local minimum of the energy function $E_{\mathbf{x}}(\mathbf{J}, \theta)$ in (2).

The *minimum probability flow* (MPF) objective function [5] given the collection \mathcal{D} is

$$K_{\mathcal{D}}(\mathbf{J}, \theta) := \sum_{\mathbf{x} \in \mathcal{D}} \sum_{\mathbf{x}' \in \mathcal{N}(\mathbf{x})} \exp\left(\frac{E_{\mathbf{x}} - E_{\mathbf{x}'}}{2}\right). \quad (4)$$

The function in (4) is infinitely differentiable and strictly convex in the parameters. Notice that when $K_{\mathcal{D}}(\mathbf{J}, \theta)$ is small, the energy differences $E_{\mathbf{x}} - E_{\mathbf{x}'}$ between $\mathbf{x} \in \mathcal{D}$ and patterns \mathbf{x}' in neighborhoods $\mathcal{N}(\mathbf{x})$ will satisfy (3), making \mathbf{x} a fixed-point of the dynamics.

As the following result explains, minimizing (4) given a storable set of patterns will determine a Hopfield network storing those patterns.

Theorem 1. *If a set of binary vectors \mathcal{D} can be stored as local minima of a Hopfield network, then minimizing the convex MPF objective (4) will find such a network.*

Proof: We first claim that \mathcal{D} can be stored as local minima of a Hopfield network \mathcal{H} if and only if the MPF objective (4) satisfies $K_{\mathcal{D}}(\mathbf{J}, \theta) < 1$ for some \mathbf{J} and θ . Suppose first that \mathcal{D} can be made strict local minima with parameters \mathbf{J} and θ . Then for each $\mathbf{x} \in \mathcal{D}$ and $\mathbf{x}' \in \mathcal{N}(\mathbf{x})$, inequality (3) holds. In particular, a uniform scaling in the parameters will make the energy differences in (4) arbitrarily large and negative, and thus K can be made less than 1. Conversely, suppose that $K_{\mathcal{D}}(\mathbf{J}, \theta) < 1$ for some choice of \mathbf{J} and θ . Then each term in the sum of positive numbers (4) is less than 1. This implies that the energy difference between each $\mathbf{x} \in \mathcal{D}$ and $\mathbf{x}' \in \mathcal{N}(\mathbf{x})$ satisfies (3). Thus, \mathcal{D} are all strict local minima.

We now explain how the claim proves the theorem. Suppose that \mathcal{D} can be stored as local minima of a Hopfield network; then, $K_{\mathcal{D}}(\mathbf{J}, \theta) < 1$ for some \mathbf{J}, θ . Any method producing parameter values \mathbf{J} and θ having objective (4) arbitrarily close to the infimum of $K_{\mathcal{D}}(\mathbf{J}, \theta)$ will produce a network with MPF objective strictly less than 1, and therefore store \mathcal{D} by above. \square

Our next main result is that at least n patterns in an n -node Hopfield network can be stored by minimizing (4). To make this statement mathematically precise, we introduce some notation. Let $r(m, n) < 1$ be the probability that a collection of m binary patterns chosen uniformly at random from all $\binom{2^n}{m}$ m -element subsets of $\{0, 1\}^n$ can be made local minima of a Hopfield network. The *pattern capacity* (per neuron) of the Hopfield network is defined to be the supremum of all real numbers $a > 0$ such that

$$\lim_{n \rightarrow \infty} r(an, n) = 1. \quad (5)$$

Theorem 2. *The pattern capacity of an n -node Hopfield network is at least 1 pattern per neuron.*

In other words, for any fixed $a < 1$, the fraction of all subsets of $m = an$ patterns that can be made strict local minima (and thus fixed-points) of a Hopfield network with n nodes converges to 1 as n tends to infinity.

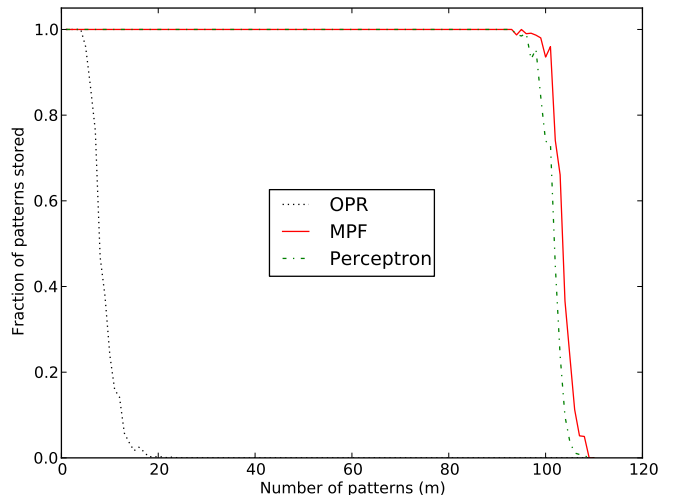


FIG. 3. Shows fraction of patterns made fixed-points of a Hopfield network using OPR (outer-product rule), MPF (minimum probability flow), and PER (perceptron) as a function of the number of randomly generated training patterns m . Here, $n = 64$ binary nodes and we have averaged over $t = 20$ trials. The slight difference in performance between MPF and PER is due to the extraordinary number of iterations required for PER to achieve perfect storage of patterns near the critical pattern capacity of the Hopfield network. See also Fig. 4.

Moreover, by Theorem 1, such networks can be found by minimizing (4). Experimental evidence suggests that the limit in (5) is 1 for all $a < 1.5$, but converges to 0 for $a > 1.7$ (see Fig. 3). Although the Cover bound [14] forces $a \leq 2$, it is an open problem to determine the exact critical value of a (i.e., the exact pattern capacity of the Hopfield network).

We close this section by defining a new learning rule for a neural network. In words, the *minimum probability flow learning rule* (MPF) takes an input training pattern \mathbf{x} and moves the parameters (\mathbf{J}, θ) a small amount in the direction of steepest descent of the MPF objective function $K_{\mathcal{D}}(\mathbf{J}, \theta)$ with $\mathcal{D} = \{\mathbf{x}\}$. Mathematically, these updates for J_{ij} and θ_i take the form (where again, $\delta = \mathbf{1} - 2\mathbf{x}$):

$$\Delta J_{ij} \propto -\delta_i x_j e^{\frac{1}{2}(\mathbf{J}_i \mathbf{x} - \theta_i) \delta_i} - \delta_j x_i e^{\frac{1}{2}(\mathbf{J}_j \mathbf{x} - \theta_j) \delta_j} \quad (6)$$

$$\Delta \theta_i \propto \delta_i e^{\frac{1}{2}(\mathbf{J}_i \mathbf{x} - \theta_i) \delta_i}. \quad (7)$$

It is clear from (6),(7) that MPF is a local learning rule.

Experimental results. We performed several experiments comparing standard techniques for fitting Hopfield networks with minimizing the MPF objective function (4). All computations were performed on standard desktop computers, and we used the limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm [15] to minimize (4).

In our first experiment, we compared MPF to the two methods OPR and PER for finding 64-node Hopfield networks storing a given set of patterns \mathcal{D} . For each of 20

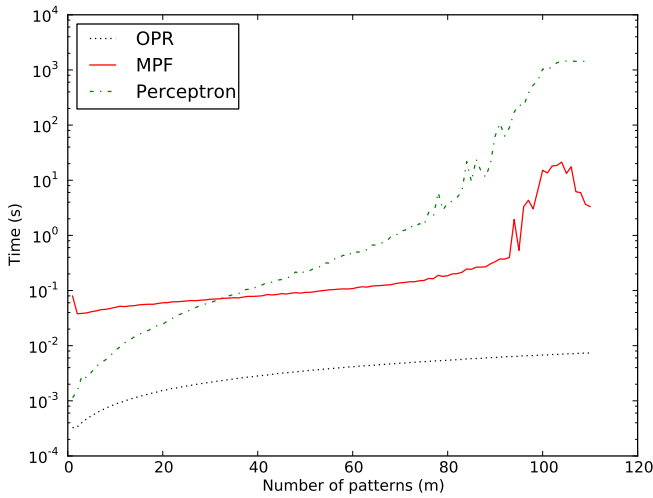


FIG. 4. Shows time (on a log scale) to train a Hopfield network with $n = 64$ neurons to store m patterns using OPR, PER, and MPF (averaged over $t = 20$ trials).

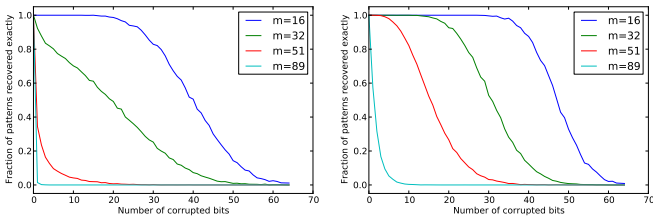


FIG. 5. Shows fraction of exact pattern recovery for a perfectly trained $n = 128$ Hopfield network using rules PER (figure on the left) and MPF (figure on the right) as a function of bit corruption at start of recovery dynamics for various numbers m of patterns to store. We remark that this experiment and the next do not involve OPR as its performance was significantly worse than that of MPF and PER.

trials, we used the three techniques to store a randomly generated set of m binary patterns, where m ranged from 1 to 120. The results are displayed in Fig. 3 and support the conclusions of Theorem 1 and Theorem 2.

To study the efficiency of our method, we compared training time of a 64-node network as in Fig. 3 with the three techniques OPR, MPF, and PER. The resulting computation times are displayed in Fig. 4 on a logarithmic scale. Notice that computation time for MPF and PER significantly increases near the pattern capacity threshold of the Hopfield network.

For our third experiment, we compared the denoising performance of MPF and PER. For each of four values for m in a 128-node Hopfield network, we determined weights and thresholds for storing all of a set of m randomly generated binary patterns using both MPF and PER. We then flipped 0 to 64 of the bits in the stored patterns and let the dynamics (1) converge (with weights and thresholds given by MPF and PER), recording if the converged pattern was identical to the original pattern or not. Our

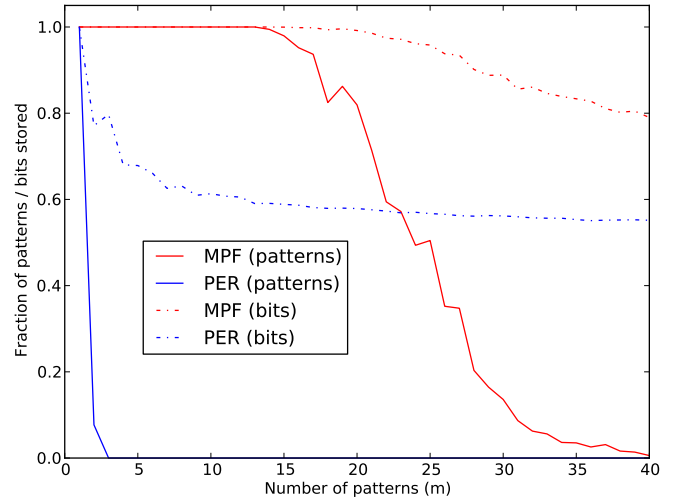


FIG. 6. Shows fraction of patterns (shown in red for MPF and blue for PER) and fraction of bits (shown in dotted red for MPF and dotted blue for PER) recalled of trained networks (with $n = 64$ nodes each) as a function of the number of patterns m to be stored. Training patterns were presented repeatedly with 20 bit corruption (i.e., 31% of the bits flipped). (averaged over $t = 13$ trials.)

results are shown in Fig 5, and they demonstrate the superior corrupted memory retrieval performance of MPF.

A surprising final finding in our investigation was that MPF can store patterns from highly corrupted or noisy versions on its own and without supervision. This result is explained in Fig 6. To illustrate the experiment visually, we stored $m = 80$ binary fingerprints in a 4096-node Hopfield network using a large set of training samples which were corrupted by flipping at random 30% of the original bits; see Fig. 2 for more details.

Discussion. We have presented a novel technique for the storage of patterns in a Hopfield associative memory. The first step of the method is to fit an Ising model using minimum probability flow learning to a discrete distribution supported equally on a set of binary target patterns. Next, we use the learned Ising model parameters to define a Hopfield network. We show that when the set of target patterns is storable, these steps result in a Hopfield network that stores all of the patterns as fixed-points. We have also demonstrated that the resulting (convex) algorithm outperforms current techniques for training Hopfield networks.

We have shown improved recovery of memories from noisy patterns and improved training speed as compared to training by PER. We have demonstrated optimal storage capacity in the noiseless case, outperforming OPR. We have also demonstrated the unsupervised storage of memories from heavily corrupted training data. Furthermore, the learning rule that results from our method is local; that is, updating the weights between two units requires only their states and feedforward input.

As MPF allows the fitting of large Hopfield networks

quickly, new investigations into the structure of Hopfield networks are possible [16]. It is our hope that the robustness and speed of this learning technique will enable practical use of Hopfield associative memories in both computational neuroscience, computer science, and scientific modeling.

Acknowledgements. This work was supported by an NSF All-Institutes Postdoctoral Fellowship administered by the Mathematical Sciences Research Institute through its core grant DMS-0441170 (CH) and by NSF grant IIS-0917342 (KK).

* chillar@msri.org;
<http://www.msri.org/people/members/chillar/>

† jascha@berkeley.edu

‡ kilian@berkeley.edu

- [1] E. Ising, *Zeitschrift für Physik* **31**, 253 (Feb. 1925)
- [2] J. Hopfield, *Proceedings of the National Academy of Sciences of the United States of America* **79**, 2554 (1982)
- [3] M. Cohen and S. Grossberg, *IEEE Transactions on Systems, Man, & Cybernetics*(1983)
- [4] G. Hinton and T. Sejnowski, *Parallel distributed processing: Explorations in the microstructure of cognition* **1**, 282 (1986)
- [5] J. Sohl-Dickstein, P. Battaglino, and M. DeWeese, *Physical Review Letters*(2011)
- [6] J. Hertz, A. Krogh, and R. Palmer, *Introduction to the theory of neural computation*, Vol. 1 (Westview press, 1991)
- [7] G. Weisbuch and F. Fogelman-Soulié, *Journal de Physique Lettres* **46**, 623 (1985), ISSN 0302-072X
- [8] R. McEliece, E. Posner, E. Rodemich, and S. Venkatesh, *Information Theory*, *IEEE Transactions on* **33**, 461 (1987), ISSN 0018-9448
- [9] D. Amit, H. Gutfreund, and H. Sompolinsky, *Annals of Physics* **173**, 30 (1987), ISSN 0003-4916
- [10] F. Rosenblatt(1957)
- [11] M. Minsky and S. Papert, *Perceptrons* (MIT press, 1988) ISBN 0262010976
- [12] M. Jinwen, in *Neural Networks, 1993. IJCNN'93-Nagoya. Proceedings of 1993 International Joint Conference on*, Vol. 3 (IEEE, 1993) pp. 2611–2614
- [13] F. Sommer and P. Dayan, *Neural Networks*, *IEEE Transactions on* **9**, 705 (1998)
- [14] T. Cover, *Electronic Computers*, *IEEE Transactions on*, 326(1965), ISSN 0367-7508
- [15] J. Nocedal, *Mathematics of computation* **35**, 773 (1980)
- [16] C. Hillar, N. Tran, and K. Koepsell(2012)